STOP PRESS

DRAGON

INFORMATION BOOM FOR USERS

Interest around the Dragon 32 Home Computer is growing at a rapid rate. To satisfy information hungry Dragon Users, a wide selection of material on how to enjoy programming this remarkable machine is becoming widely available.

The first edition of our newsletter was enthusiastically received and many interesting letters have been received by the editor. These include some exciting new program listings and examples will be included in the next issue of the newsletter.

We have expanded this issue of the newsletter to an 8 page format and if the response continues to grow there are plans to develop the newsletter into a regular magazine style publication.

An independent Dragon User magazine has just been produced by Sunshine Publications and will appear in the news agencies this month with a cover price of 60p. We have enclosed the first issue free of charge with the compliments of Dragon Data Ltd.

Several books are appearing in the shops on the Dragon 32. Notable among these are "The Dragon 32" by Ian Sinclair (published by Granada and priced at £5.95) and "The Working Dragon 32" by David Lawrence (published by Sunshine Books Ltd. priced at £5.95). Other books are planned including one by Penguin. Watch out for "Dragon Magic" by Foulsham Publications, which will be aimed at the younger user and should be published shortly.

The Dragon Users Club
As mentioned before all Users who have returned their guarantee cards for the Dragon 32 to Dragon Data Ltd. are automatically members of the Dragon Users Club, which entitles them to free issues of the Dragon Newsletter

Many local or regionally based Dragon User Clubs are being established around the country by Dragon enthusiasts wishing to share their experiences with other users who live nearby. If you have started such a club or are already a member of one of them we at Dragon Data would be delighted to hear from you.

We would like to publish a complete list of such clubs in a future issue of the newsletter. Please therefore write to:

Miss Cathy Hyde
Dragon Data Ltd.
Kenfig Industrial Estate
Margam
Port Talbot
West Glamorgan

In this way we can put other users in your area in touch with your club. If there does not appear to be a club in your area then write to Cathy at Dragon and she will help you find a suitable club.

Dragon Data do not intend to involve themselves directly in the running of local clubs, believing it best left to the enthusiasm of their members. However, if club officials would like support material for User's Days etc. then write to Cathy for assistance.

Contributions
If you would like to contribute programs, comments, suggestions, hints or stories on using the Dragon 32 then please write enclosing details to:

The Editor, The Newsletter, c/o Dragon Data Ltd., Kenfig Industrial Estate, Margam, Port Talbot, West Glamorgan.

All program suggestions should enclose a program listing and a letter explaining its basic structure. Long programs should be submitted on a cassette tape which is itself clearly labelled with your name, and an addressed envelope for its return. List program lines with a maximum of 64 characters per line (two screen lines) and number lines in multiples of 10.

MACHINE CODE CORNER

The 6809 microprocessor, which is the heart (or perhaps we should say "brain") of the Dragon, is an extremely flexible device which allows you, the programmer, a lot more power than is available from other microprocessors. There are nine temporary storage registers, and nearly 1500 different instructions with which to manipulate them. We won't discuss them all in this article!

Let's start with just four of the registers: X, Y, A and B. X and Y are both 2-byte (16-bit) *index registers*, which can store the same amount of information as 2 bytes of memory – that is any number from 0 to 65535. In particular, they can store the "address" of any of the Dragon's memory locations. A and B are 1-byte (8-bit) *accumulators* which are used for manipulating data. In Assembly language A and B are the nearest we have to an ordinary BASIC variable with which we can perform the usual arithmetic operations like adding and subtracting. A and B may be combined to form D, a 2-byte accumulator, A being the most significant byte.

An Assembly language program is written in four columns or *fields*. Each statement takes one line, but need not use all four fields. The first field is the Symbol field. One use of this is to label a line for branching purposes. The second field is the Command field, in which the main statement is placed. The third is the Operand, which may be a memory address or data. The fourth is for Comments: the equivalent of BASIC REM's.

Some useful commands are those which let us put numbers into the registers – the equivalent of "**LET A = 5**" or just "**A = 5**" in BASIC. In Assembly language, this becomes

```
LDA   #5
```

The command is LDA (load A) and the operand is #5, which just means the *number 5*. This type of operand is called Immediate Addressing – the operand is the data itself, as shown by the # sign.

Another form of the load command is

```
LDA   $7FFF
```

(The $ shows it is a hexadecimal number.) This loads A with the contents of memory $7FFF, not the actual number $7FFF (which is too big for A anyway). This is called Extended Addressing – the operand is an address.

A third form of the LDA command is

```
LDA   ,X
```

This loads A with the contents of the memory indicated by the X register. So if X contains the value $400, A will be loaded with the contents of memory $400. This is called Indexed Addressing, and is indicated by the comma, in the same way that Immediate is indicated by #.

There are load commands for the other registers: LDB, LDD, LDX, LDY. These have basically the same effect, except that the 2-byte registers, when loaded with Extended or Indexed Addressing, take the 2-byte value which is made up of the memory indicated and the memory which follows it. So in the case of Indexed Addressing,

```
LDD   ,X
```

(where X has the value $400, $400 contains the value $12, and $401 contains the value $34) will load D with the value $1234.

Having loaded a register, it is useful to be able to store the value somewhere. This is done with the store commands, STA, STB, STD, STX, STY. For example,

```
STA   $7FFF
```

stores the contents of register A in memory $7FFF (Extended Addressing). Similarly,

```
STY   ,X
```

stores the most significant byte of the contents of register Y in the memory indicated by register X, and the other byte in the memory following (Indexed Addressing).

A useful facility available with Indexed Addressing is the auto-increment and auto-decrement. For example

```
LDA   ,X+
```

loads A from the memory indicated by X, then adds 1 to the value of X. Auto-decrement works in reverse order:

```
LDA   ,-X
```

first subtracts 1 from X, then loads A from the memory indexed by X.

A third set of commands which are very useful is the set of compare commands, CMPA, CMPB, CMPD, CMPX, CMPY. These may be followed by conditional branching commands. For example

```
CMPX   #$1E00
BNE    LOOP
```

compares X with the number $1E00, and "branches if not equal" (BNE) to the statement labelled LOOP.

There are more registers, more forms of address and of course many more commands, but we have enough now to write a simple program. The area of memory used by the high resolution graphics commands in PMODE3,1 and PMODE4,1 is $600 to $1E00. The following program fills the graphics screen with the configuration represented by the byte $7FFF:

```
        LDA    $7FFF
        LDX    #$600
LOOP    STA    ,X+
        CMPX   #$1E00
        BNE    LOOP
        RTS
```

If you have an assembler, you will be able to assemble this and run it immediately. But those of you who want to translate it yourselves need to know some "opcodes". The opcode for any command depends on the addressing mode. The ones discussed in this article are, in hexadecimal:

	LDA	LDB	LDD	LDX	LDY	STA	STB	STD	STX	STY
Immediate	86	C6	CC	8E	108E	-	-	-	-	-
Extended	B6	F6	FC	BE	10BE	B7	F7	FD	BF	10DF
Indexed	AC	E6	EC	AE	10AE	A7	E7	ED	AF	10AF

10 BF

	CMPA	CMPB	CMPD	CMPX	CMPY
Immediate	81	C1	1083	8C	108C
Extended	B1	F1	10B3	BC	10BC
Indexed	A1	E1	10A3	AC	10AC

As you see, some commands take 2 bytes. The opcode for BNE is $26. Its addressing mode is *relative*, which means that its operand is worked out relative to the current position (which is the first byte of the next instruction). RTS has opcode $39. The operand ,X+ is repesented by $80.

The program now becomes, line by line,

```
B6 7F FF, 8E 06 00, A7 80, 8C 1E 00, 26 F9, 39
```

or, in decimal,

```
182 127 255, 142 6 0, 167 128, 140 30 0, 38 249, 57.
```

It can be entered into the Dragon, starting at byte 32000, using the BASIC program:

```
10 FOR I=0 TO 13 : READX : POKE 32000+I,X :
   NEXTI
20 DATA182,127,255,142,6,0,167,128,140,30,0,38,
   249,57
```

To investigate the available graphic "effects", try running the BASIC program:

```
10 PCLEAR4 : PMODE3 : SCREEN1,0
20 FOR I=0 TO 255 : POKE &H7FFF ,I :
   EXEC32000
30 FOR J=0 TO 1000 : NEXTJ,I
```

When the value in $7FFF is 0, 85, 170 or 255, the result is the same as using the PCLS command.

You may like to experiment with minor alterations in the machine code. For example, if you alter the 5th byte from 6 to 18 (making the 2nd line of the program LDX #$1200) the result is an operation on the lower half of the screen only. Similarly, the 10th byte may be adjusted to alter the lower limit of the effect. Keep experimenting - it's the best way to learn about your computer.

ERRATA

Many of the letters to the editor received in response to the issue of the newsletter pointed out that there were some errors in the program listings.

Thank you to Mr. Alan Jones of Banbury who wrote to say that he found the newsletter "most useful" but noticed that line 20 of the Machine code program featured a Z instead of a 0, Mr. J. Johnson of Potters Bar who found an error in the Cassette Loading article, which we have reprinted correctly in our follow-up feature on page 5 of this issue, and to Mr. O'Mulloy of Cambridge, who amongst many others noted that the spaces were missing in the Dragon Bytes examples.

Like many other publishers before us we have found that taking program listings to the typesetting stage can cause problems. We have, therefore, listed below some corrections which we hope might help a number of confused users around the country:

1. Dragon Bytes-examples

```
(a) FORI=X  TO5      (b) IFX=Y  THENSTOP
 (c) ONX  GOTO1,2,3
```

2. What are Hexadecimal numbers? - Colon instead of semi-colon in line 40

```
40 RE=X-16*Y:GOSUB100:
   H$=A$+H$: IFY=0THEN 60
```

Machine Code corner-second line of program has a Z instead of 0

```
20  FOR I=0 TO 13:READ J:POKE
    I+32000,J:NEXT I
```

Also, in the assembly language program two #'s are missing-

```
2 LDX        #$400
4 CMPX       #$600
```

MORE DRAGON BYTES

How much of Dragon's 32K of RAM is actually available to us for BASIC programs? Try switching your Dragon on and typing

```
?MEM
```

Back will come the answer: 24871. So is this all our Dragon really has to offer us – less than 25K? A few ill-informed writers have claimed that to be the case. But they are wrong.

When you switch on, certain areas of memory are automatically reserved – 200 bytes for string storage and 6144 bytes for high resolution graphics. Since the remaining 24871 bytes are ample for most purposes, we don't usually bother to alter these default reservations, even if there are no strings or graphics in our program. But the extra bytes *are* available.

First type **CLEAR 0**
This releases the 200 string bytes. ?MEM now gives a value of 25071 bytes.

Now to release the high resolution graphics bytes: type

```
PCLEAR 1
```

This releases all but one of the graphics "pages". ?MEM now gives 29679. Unfortunately we cannot use the command PCLEAR 0, but the effect can be achieved by typing in the two commands:

```
POKE25,6
NEW
```

Now ?MEM gives 31215 bytes. All of this memory is now available for our BASIC program – but of course we can't use high resolution graphics.

We *can* make limited use of strings even without any memory reserved for string storage. A statement like

```
X$ = "THIS IS A STRING"
```

doesn't use any string space. Neither does

```
10 DATA"THIS IS A STRING"
20 READ X$
```

The statement

```
Y$ = X$
```

doesn't make a *new* string (only a copy of one that already exists – X$) and so doesn't use any string space.

But statements like

```
Z$ = X$ + Y$   and
A$ = LEFT$(B$,1)
```

5FF

make new strings and therefore need string space to store them.

The INPUT statements

```
INPUT X$   and
INPUT #-1,X$
```

also need string space to store the strings which are input.

So using the methods described above, we can make 31215 bytes available for BASIC use, and even with no string space reserved we can make limited use of string variables.

ORDER FROM CHAOS?

a simple game with numbers

Suppose you are presented with the numbers 1 to 9 in the order 456712389, and you want to achieve their natural ordering but you are only allowed to reverse the order of the first i digits (i being at your choice). By inspection, we can proceed to reverse 4567 giving 765412389, followed by reversing 7654123 to give 321456789 and finally reversing 321 gives 123456789 - home in 3 reversals!

Now try keying in the short program below and experimenting with the (random) orderings of the nine numbers. All you have to do is to key in the number i on each occasion (no ENTER is required) so that in the example above press the keys 4 (to reverse 4567), followed by 7 (to reverse 7654123) and then 3 (to reverse 321).

Can you develop rules of thumb which use an average of ten or fewer moves?

```
  1 REM ORDER GAME
  2 REM A.M.SYKES 1983
 10 DIM A(9):FOR I=1 TO 9:A(I)=I:NEXTI:A$="123456789":
     C=0
 20 CLS:FOR I=9 TO 1 STEP-1:Z=RND (I):W =A(Z):
     A(Z)=A(I): A(I)=W:NEXTI
 30 PRINT@32,"NUMBER TO REVERSE?":PRINT@56,
     "TRIALS":GOSUB200
 40 KS=INKEYS:IF KS = "" THEN40
 50 K = INSTR(1,A$,KS): IF K=0 THEN40
 60 R=VAL(KS): C=C+1: GOSUB100: GOSUB200
 70 FOR I=1 TO 9: IF A(I)=I THEN NEXTI ELSE 90
 80 GOSUB300: GOTO20
 90 GOTO40
100 PRINT@100,R: PRINT@122,C: FOR I=1 TO INT(R/2):
     Z=A(I)
110 A(I) =A(R-I+1):A(R-I+1) = Z: NEXTI: RETURN
200 FOR I=1 TO 9: PRINT@228+2*I,A(I);: NEXTI: RETURN
300 PRINT@391,"CONGRATULATIONS"
310 PRINT@423,"YOU TOOK";C;"MOVES"
320 PRINT@450,"PRESS A KEY FOR ANOTHER GO"
330 KS = INKEY$: IF K$="" THEN 330 ELSE RETURN
```

PROMOTION

this program shows you how to get to the top!

```
  1 ' ::HOW TO GET TO THE TOP::
  2 ' MAVIS PEARSON. MARCH 1983
 10 CLS: PRINT@330,"WAIT":PCLEAR4
 20 DIM A(40,50),B(40,50):PMODE3,1:PCLS
 30 L$="ND10L35ND10R35U10"
 40 DRAW"BM 36,80;C3;"
 50 FOR I=1 TO 9: DRAW L$:NEXT
 60 CLS:PRINT@325,"FOR THE RIGHT MOMENT"
 70 B$="C4D3L4D20R10U20L4U3"
 80 CL$="BD3L5H10NU10BR20BD10F10NU10BG10
     D25R4L4U25L20D15H4"
 90 CR$="BD3R5E10NU10BL20BD10G10NU10BF10
     D25L4R4U25R20D15E4"
100 CIRCLE(20,10),6,4:PAINT(20,10),4
110 DRAW "BM18,15;C4"+BS+CL$:PAINT(20,20),4
120 GET(0,0)-(40,75),A,G:PCLS
130 DRAW"BM 36,80;C3":FOR I=1 TO 9:DRAW L$:NEXT
140 CIRCLE(17,10),6,4:PAINT(17,10),4
150 CLS:PRINT@330,"THEN"
160 DRAW"BM16,15;C4;"+B$+CRS:PAINT(16,25),4
170 GET(0,0)-(40,75),B,G:PCLS:SCREEN1,1
180 DRAW"BM138,170;C3;":FOR I=1 TO 18: DRAW L$:NEXT
190 FOR I=1 TO 9 STEP 2
200 PUT(102,100-I*10)-(142,175-I*10),A,PSET
210 PUT(102,90-I*10)-(142,165-I*10),B,PSET:NEXT
220 GOTO220
```

For a meteoric rise change line 30 to the one below.

```
30 DIMA(40,50),B(40,50),PMODE1,1:PCLS
```

Clue to puzzle on page 7

Add these lines to your program-

```
100 FOR I=3 TO 10:P=PEEK(1032+32*I)
110 PRINT@32*I+8,CHR$(P+32);:NEXT:
     PRINT@350,""
```

HOW GOOD IS YOUR CHESS?

Ralph Cook

The explosion in home micro-computers has been accompanied by the inevitable proliferation of computer games of the 'shoot everything in sight' variety. How welcome it is to see Dragon Data marketing a chess cartridge (Cyrus) which is an invaluable aid to a chess player without a readily available partner.

The instructions provided with the cartridge are clear and comprehensive. Insertion of the cartridge (with Dragon switched off!) sets up a well displayed high-resolution graphics chess board with a cursor positioned at the bottom left-hand corner. Moves are made by positioning the cursor on the piece to be moved followed by ENTER, and then repeating this procedure for the position it is to be moved to. This works well especially for beginners who are particularly helped as the proposed move is accentuated by flashing new position for old, old for new, and then finally the move. Illegal moves cause a beep.

At any stage of the game previous moves may be reviewed by pressing the space bar which reveals the Message Board and also allows you to choose from a list of commands such as H for a Hint. The message board also indicates check clearly.

The player has a choice of nine levels of competence denoted by 1 to 9. Beginners are often defeated at levels 1 and 2 and still come back for more. Levels 4 and 5 are good for the weaker club player who has to be on the lookout for traps. At level 8 it gives a reasonable game to an experienced club player of about 1900 grading, whilst the top level is of more use to postal chess players as unusual moves are often given.

The program plays a strong game in the middle but is weakest at the end game. This is a well known problem with human chess players also! The cartridge allows you to set up positions easily. Level 9 will solve all mate-in-two problems in a reasonable time but mate-in-three problems take far too long to worry about. What I wonder will happen to problem setters and solvers in the future!

All in all, Cyrus Chess Cartridge is good value for money offering a wide range of options and a partner for chess players from beginners to club players. It is a pity that no positive warning is given of a check but nothing's ever perfect is it!

Ralph Cook is a keen chess player of county standard and plays for West Glamorgan when time permits!

SPIRALS

Mr Alan Jones of Banbury sent us this adaptation of one of the Manual programs. The display of graphics lasts about eighteen minutes.

```
 10 N=30:A=15:F=1
 20 PMODE4,1:SCREEN1,1:PCLS5:COLOR 0,5
 30 LINE(0,0)-(128,96),PRESET
 40 FOR I=1 TO 1000
 50 X=X+L*SIN(R):Y=Y+L*COS(R)
 60 IF X<-128 OR X>128 THEN 110
 70 IF Y<-96 OR Y>95 THEN 110
 80 LINE-(X+128,Y+96),PSET
 90 R1=R1+N:R=R1/57.29578:L=L+0.5
100 NEXTI
110 X=0:Y=0:R=0:R1=0:N=N+A:L=0
120 IF N>165 THEN N=30:F=F+1
130 IF F/2=INT(F/2)THEN A=12 ELSE A=15
140 GOTO20
```

A further modification will change the monochrome display into colour. Change the PMODE in line 20 to PMODE3,1 and change line 80 to read

```
 80 COLOR RND(3)+1:LINE-(X+128,Y+96),PSET
```

THE CONTRIBUTORS

The editor of the Dragon Newsletter would like to thank the contributors for all their help in creating the first two issues of the Dragon newsletter.

They are Alan Sykes, Mavis Pearson and Alan Mayer, all from University College, Swansea.

EXTRA EDIT COMMANDS

Have you ever been editing a long, complicated line of a program and accidentally used the H or K commands, deleting a lot of hard work by mistake? If you have, or if you think it just might happen in the future, you will be glad to know about a few extra commands, not included in the list on page 40 of your Dragon handbook.

A abandons the current attempt to edit, restoring the line to the situation before the editor was called. You remain in EDIT mode.
E has the same effect as [ENTER], leaving the editor, storing the line and returning to keyboard.
Q has the combined effect of A and E.

These commands will work in the ordinary EDIT mode. If you have typed H you will be in insert mode, from which you must return by typing [SHIFT][↑] before you can use A or Q. If you have typed K, you must type another character to complete the "kill" sequence, before you can use A or Q to restore the line.

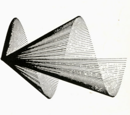
A FORETASTE OF LINES TO COME?

The use of the LINE command in high resolution graphics is worthy of an article of its own. The following program uses this command in conjunction with the SIN function.

```
10 INPUT"AMPLITUDE(80), PERIOD(4), STEP(1 OR 2)";A,P,S
20 B=P/100:PMODE3,1:SCREEN1,1:PCLS1
30 FOR I=0 TO 250 STEP S:COLORI/84+2:Y=A*SIN(B*I)
40 LINE(0,90)-(I,90+Y),PSET:NEXT I
50 GOTO50
```

The figures in brackets are suggested input values but try others.

If you would like the diagram in one colour only, edit line 30 and change the second I to 0



CASSETTE LOADING

In the last issue there was an error (a misplaced colon) in the one line command which was suggested to cure problems with Automatic Recording levels.

The line should have read:

SOUND 120,20: FOR N=1 TO 100:NEXT:CSAVE"program name"

More recording tips (1) Cassette tapes do get worn with use especially at the beginning of the tape. This can cause loading errors. It is a good idea to keep a 'library' copy of your program on a separate tape.

(2) The SKIPF command provides a method of checking that a program has been successfully saved. After using CSAVE"program name", rewind the tape and press only the PLAY on the recorder. Then enter SKIPF"program name". If your Dragon returns with OK without an IO ERROR you are very unlikely to encounter difficulties loading that program in future. If however, you do get an IO ERROR then your program is still safe in Dragon. Check for causes of the error and then try again.

HOW NOT TO GET TIED UP IN KNOTS WITH STRINGS

Have you explored Dragon's string-handling capabilities yet? If not, you may be surprised to discover how powerful they are.

The first thing you will find is that operations with strings are very quick. To demonstrate this let's take another look at the subject of the machine code demonstration in the first issue of this Newsletter, which filled the text screen with characters of one kind. A plain man's approach to this in BASIC would be to use the PRINT@ facility. The following program will suffice:

```
10  FOR I=0 TO 511:PRINT@I,"A";: NEXTI
20  GOTO20
```

You will find that the last line of A's is printed and then seems to be rubbed out! This is because the automatic "screen scroll" comes into play immediately the last character of the last line is printed. Annoying though this is, by changing line 20 to 20 GOTO10, we can count the occurrences of the bottom line and so time the printing process. Result: 2.2 seconds to fill the screen.

Can this be improved upon without recourse to Machine Code?

Yes it can, by using the BASIC command A$ = STRING$(255,"A") which assembles 225 A's in one long string (of maximum length).

```
10  CLEAR1000
20  A$ = STRING$(255,"A")
30  PRINT@0,A$;:PRINT@256,A$;
40  CLS:GOTO20
```

The resulting flashes (corresponding to the CLS operation in line 40) allow another timing of the print process. Result: approximately one sixth of a second, which is over thirteen times faster than our first attempt.

So much for speed. Add to it the use of INKEY$ to monitor what is happening at the keyboard and you have the power to construct programs which respond (instantaneously, it seems) to instructions. For example, modifying our last program to

```
10 CLEAR1000
20 K$=INKEY$ : IF K$="" THEN 20
30 A$=STRING$(255,80+ASC(K$))
40 PRINT@0,A$;:PRINT@255,A$;
50 GOTO20
```

and we have a BASIC program which seems almost as quick as its machine code equivalent.

The command INKEY$ is also very useful for the control of printout that causes the screen to scroll. A more sophisticated alternative to the BREAK key followed by CONT is demonstrated below. Press any key to stop and then restart the program.

```
10 A=A+1 : PRINT A
20 A$=INKEY$ : IF A$="" THEN40
30 A$=INKEY$ : IF A$="" THEN30
40 GOTO10
```

Many programs offer the user a choice and the common way to present that choice is with a MENU. The following program demonstrates how this can be done painlessly and routinely.

```
10 A$="ABC" : CLS
20 PRINT@33, "YOU HAVE A CHOICE - TYPE IN"
30 PRINT@170,"A...FOR APPLE"
40 PRINT@234,"B...FOR BANANA"
50 PRINT@298,"C...FOR CURRANT"
60 C$=INKEY$ : IF C$="" THEN60
70 C=INSTR(1,A$,C$) : IF C=0 THEN 60
80 ON C GOSUB90,100,110 : GOTO20
90 PRINT@490,"APPLE      ";: RETURN
100 PRINT@490,"BANANA   ";: RETURN
110 PRINT@490,"CURRANT";: RETURN
```

The crux of the program lies in the use of the function INSTR in line 70 which searches through string A$ starting with character 1 to find C$. It records the position of the first occurrence of C$, or 0 if C$ is not present. Connect that to line 80 which uses ON...GOSUB and you have an extremely neat method of branching to alternative procedures at the touch of the appropriate key!

There are numerous ways to exploit this technique - it is particularly useful in conjunction with DRAGON's PLAY and DRAW commands. This final program allows you to key in PLAY commands and then to hear the result:

```
10 A$=CHR$(8)+";"+CHR$(13) : CLS : F$="" :
T$=""
20 K$=INKEY$ : IF K$="" THEN20
30 K=INSTR(1,A$,K$) : IF K=0 THEN T$=T$+K$
: GOSUB150
40 ON K GOSUB100,110,120 : GOTO20
100 T$=LEFT$(T$,LEN(T$)-1) : GOSUB150 :
    RETURN
110 T$=T$+";" : F$=F$+T$ : T$="" :
    GOSUB200 : RETURN
120 GOSUB110 : PLAY F$ : F$="": CLS :
    GOSUB200 : RETURN
150 PRINT@32,T$ : RETURN
200PRINT@192,F$ : T$="" : GOSUB150 :
    RETURN
```

To try the program out key in PLAY commands. (Illegal ones will be accepted but will produce "FC ERROR IN 120"). The program is so constructed as to accept segments of the tune (up to a semi-colon) and then add the latest segment to the full play command. The backward arrow may be used to correct mistakes (prior to the semi-colon) and the Play command is executed on pressing ENTER.

With minor modifications the program can be adapted to play the tune backwards as well! With so many Strings to one's bow the possibilities are endless!

YOUNG USERS PAGE

DRAGON PUZZLE 1

Can you fill in the missing words in the program below? Then type and run it to find the hidden word.

```
10  CLS:PRINT@10,"DRAGON PUZZLE 1"
20  PRINT@294,"...."         Dragon's breath
30  PRINT@263,"....."        Chess, tennis, rugby
                             are all...
40  PRINT@132,"......"       Pooh's companion
50  PRINT@196,"........"     A team game
60  PRINT@167,"......"       This creature has a powerful
                             hug
70  PRINT@325,"....."        This black and white
                             creature looks huggable
80  PRINT@230,"......"       The saint dragons don't like
90  PRINT@101,"....."        Slow down quickly
100 GOTO100
```

Have you found the hidden word? Look on page 4 for a clue.

LIQUORICE ALLSORTS

Q. When does a TV screen look like a bag of liquorice allsorts?

A. When you use some of the lower graphics characters.

You can print *all sorts* of symbols on the screen by using their code numbers – you'll find them on pages 136-138 of the Dragon Programming Manual.

Wake your Dragon up and type

PRINT CHR$(153) followed by ENTER.

There you have a yellow and black allsort. You can put it anywhere you like on the screen by using the PRINT@ command and a position number from 0 to 511 (see page 140 of the manual).

Type in **PRINT@300,CHR$(153)**

Try different numbers and see what happens. Your Dragon will growl an error if you step on its tail by using a number that is too big.

Here is a Liquorice Allsorts program. It shows you all the symbols you can get. Try typing it in.

```
10  CLS                  clears the screen
20  K=32                 sets first character value
30  I=2*K                sets position for printing
40  PRINT@I,CHR$(K);     prints character K at position I
50  K=K+1                increases the value of K for
                         next character
60  IF K<256 THEN 30     takes the program through all
                         values of K
RUN
```

Have you ever played that game of trying to go from one word to another changing one letter at a time? e.g. scot, soot, foot, fool, foal, coal. . . . Each word on the way must be a real word. Well we can change this program line by line and run it after each change and get a different effect:

Change line 20 to

```
20 K=RND(255)
RUN
```

This line selects a random starting place for K from 1 to 255. Now let's print all over the place with a new line 30:

```
30 I=RND(500)
RUN
```

So far we've been going through the symbols in order but starting at a random point. By changing line 50 we can put random characters in random positions.

```
50 GO TO 20
RUN
```

You will have to press the BREAK key to stop this one. Lastly we can use a special function which puts together words of the same character. Try typing

```
PRINT STRING$(32,42)
```

Do you see stars before your eyes?

So our new line 40 is

```
40 PRINT@I ,STRING$(RND(32),K);
RUN
```

Well now we have lots of liquorice allsorts and all sorts of other things. There's still room for change. Try taking the semi-collons out of the print line or a new line 20 K = RND(128)+127 or line 30 I = RND(480)

Keep experimenting. You can do ALL SORTS of things.

LEARN YOUR TABLES

Here is a little program to test your mental arithmetic. As written here, it gives you 10 multiplication problems. Both numbers are chosen at random between 1 and 12. You have about 4 seconds to key in the right answer otherwise it is counted as wrong. Your score is indicated in the bottom left hand corner of the screen.

```
 10  REM MULTIPLY
 20  REM A.D.MAYER 1983
 30  CLS:Q=0:R=0
 40  PRINT@42,"MULITPLY"
 50  PRINT@390,"OUT OF": PRINT@173,"BY"
 60  FOR I=0 TO 10: PRINT@328,R
 70  PRINT@456,Q: IF I=10 THEN STOP
 80  X=RND(12): Y=RND(12): TIMER=0
 90  PRINT@108,X: PRINT@236,Y: Z$=""
100   X$=INKEY$: Z$=Z$+X$
110  PRINT@432,Z$
120  IF VAL(Z$) = X*Y THEN 150
130  IF TIMER<200 THEN 100
140  Q=Q+1: NEXTI
150  Q=Q+1: R=R+1: NEXTI
```

If you find it a bit easy – try a few modifications. The 4 second limit can be decreased by reducing the 200 in line 130. The range of numbers can be increased by altering the 12's in line 80. The number of questions is controlled by the 10's in lines 60 and 70. You might like to replace the multiply function in line 120 by add, subtract or divide (in which case you would, of course, modify some of the words). By the way, although the backspace appears to work, you *don't* get a second chance!

COMPETITION

WIN 4 SOFTWARE CASSETTES

If you are twelve years old or under and would like to win some Dragon Software then devise a program on your Dragon 32 that uses the computer's graphic capabilities to draw the Dragon logo featured throughout this newsletter. The program should be between 10 and 20 lines.

The young programmer judged by our contributors to have devised the neatest program will win 4 cassettes of their choice from the software range listed on the back page and some posters illustrating some popular Dragon games.

Send your entries to The Editor of the newsletter at Dragon Data. You can find the address on the front page of this issue.

DRAGON
SOFTWARE

SOFTWARE AVAILABLE FOR THE DRAGON 32

A.0500	**DRAGON SELECTION TWO**	Four games for the younger user. Written in BASIC, they can be listed and edited.
A.0501	**DRAGON SELECTION TWO**	Collection of utilities. Create your own data base, write your own tunes.
A.0502	**QUEST**	Adventure game in amedieval setting. Defeat Moorlock, master of the dark castle.
A.0503	**MADNESS AND THE MINOTAUR**	A real-time adult adventure game.
A.0504	**PERSONAL FINANCE**	Keep track of family finances.
A.0505	**GRAPHIC ANIMATOR**	Create simple cartoons on the screen and animate them by flipping through the pages. Joysticks required.
A.0506	**COMPUTAVOICE**	Your Dragon will talk with this voice synthesizer.
A.0507	**EXAMPLES FROM THE MANUAL**	30 examples from the programming manual.
A.0508	**CALIXTO ISLAND**	An adventure game. Return the hidden treasure to its rightful place.
A.0509	**BLACK SANCTUM**	An adventure game. Overcome the forces of blach magic.
A.0512	**TYPING TUTOR**	Improve your speech and accuracy.
A.0513	**DRAGON MOUNTAIN**	An adventure game. Defeat the guardians of the treasure hidden in the mountain.
A.0514	**FLAG**	Race your opponent through a constantly changing maze to the final flag. Joysticks required.
A.0518	**EL DIABLERO**	An adventure game set in the desert.
A.0100	**BERSERK**	A challenging shooting game, based on the popular arcade game, one or two players. A high resolution game in black and white. Joysticks requires.
A.0101	**METEOROIDS**	Guide your ship through treacherous asteroid belt. A game requiring skill, fast reactions and concentration. A high resolution game in black and white. Joysticks optional.
A.0102	**COSMIC INVADERS**	Dragon version of the famous arcade game.
A.0103	**GHOST ATTACK**	Maze game for one player. Joysticks required.
A.0104	**CAVE HUNTER**	Descend into the maze of caves in search of gold. Joysticks required.
A.0106	**STARSHIP CHAMELEON**	Protect your planet from the attacks of the Gabulators. High quality arcade game with superb graphics and sound. Joysticks required.
A.0107	**ASTROBLAST**	Defend your ship against waves of attackers. A high resolution game in black and white. Joysticks reqired.
A.0108	**CHESS**	Nine levels of play, from beginner to master.
A.0111	**RAIL RUNNER**	Move Bill Switchman across the tracks, avoiding trains, to rescue Herman Hobo.

Published by: Welbeck Public Relations, 2 Endell Street, London WC2. Printed by: Drydens Printers Ltd., Brent Crescent, North Circular Road, London NW10.